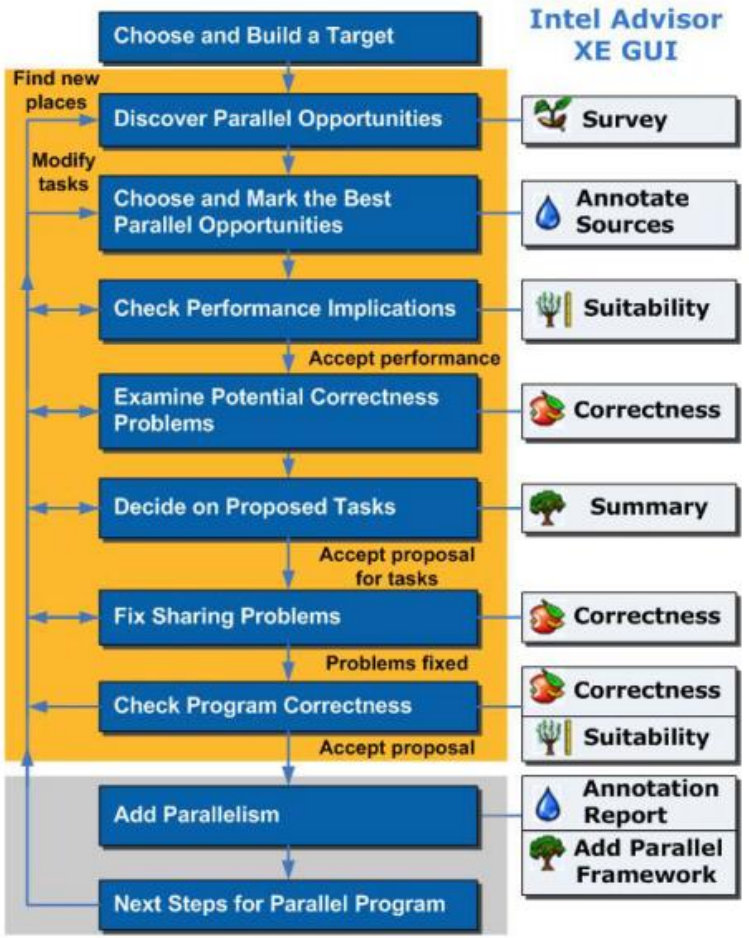
Intel® Advisor XE 2013

Intel® Advisor XE

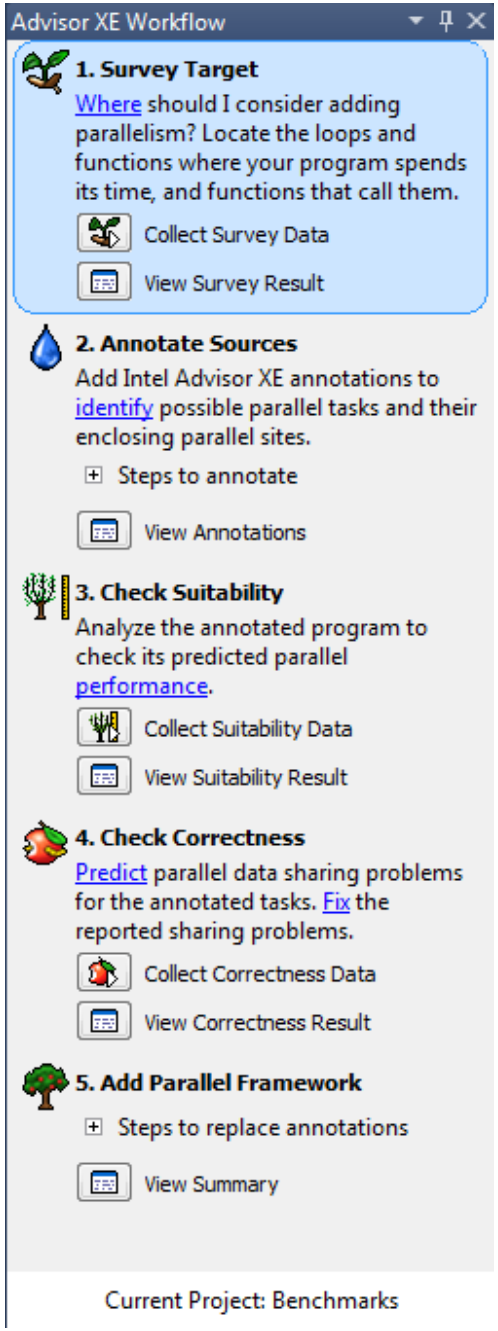
- Tool for what-if analysis
  - Modeling: use code annotations to introduce parallelism
  - Evaluation: estimate the effect e.g. the speedup
  - GUI-driven assistant (5 steps)
- Productivity and Safety
  - Parallel correctness is checked based on a correct program
  - Non-intrusive API
- It's not auto-parallelization
- It's not modifying the code



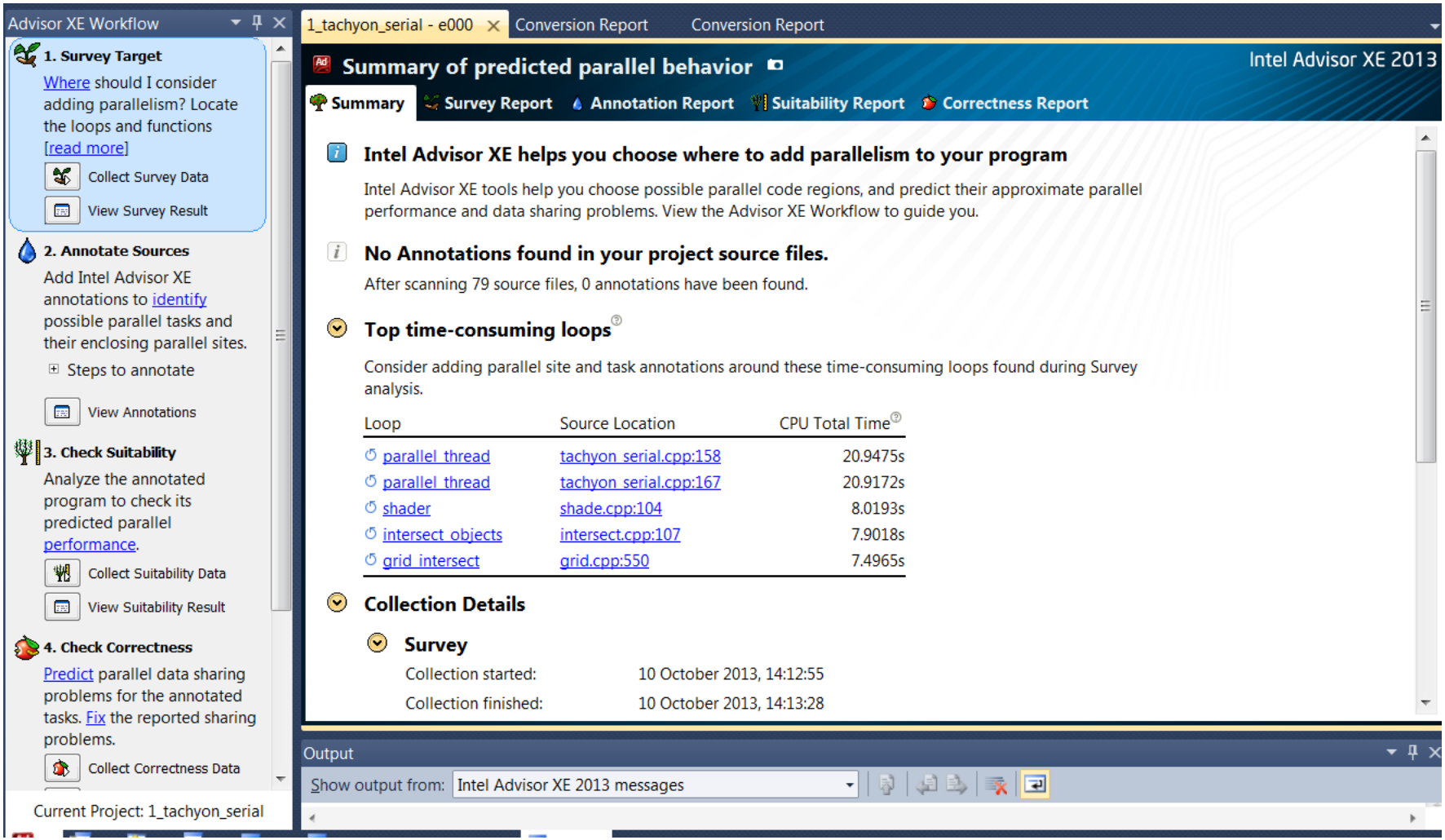
Intel® Advisor XE Stages

Transforming many serial algorithms into parallel form takes 5 easy high-level steps:

1. **Survey and Summary tools**: where to add parallelism
2. **Annotations**: experiment with parallel program structure
3. **Suitability tool**: predict and model program scalability
4. **Correctness tool**: discover potential synchronization problems
5. Manually convert annotations to **parallel framework** API (with a little help of Annotations/Summary)



Survey Target



Annotate Sources

What are annotations?

- Macros (function calls) that can be easily disabled*
- Inform analysis tool about intended parallelization

Example Intel Advisor annotations:

```
#define ANNOTATE_SITE_BEGIN(NAME)      call annotate_site_begin(NAME)
#define ANNOTATE_SITE_END(NAME)        call annotate_site_end()
#define ANNOTATE_ITERATION_TASK(NAME)  call annotate_iteration_task(NAME)
#define ANNOTATE_TASK_BEGIN(NAME)      call annotate_task_begin(NAME)
#define ANNOTATE_TASK_END(NAME)        call annotate_task_end()
#define ANNOTATE_LOCK_ACQUIRE(ADDRESS) call
annotate_lock_acquire(ADDRESS)
#define ANNOTATE_LOCK_RELEASE(ADDRESS) call
annotate_lock_release(ADDRESS)
```

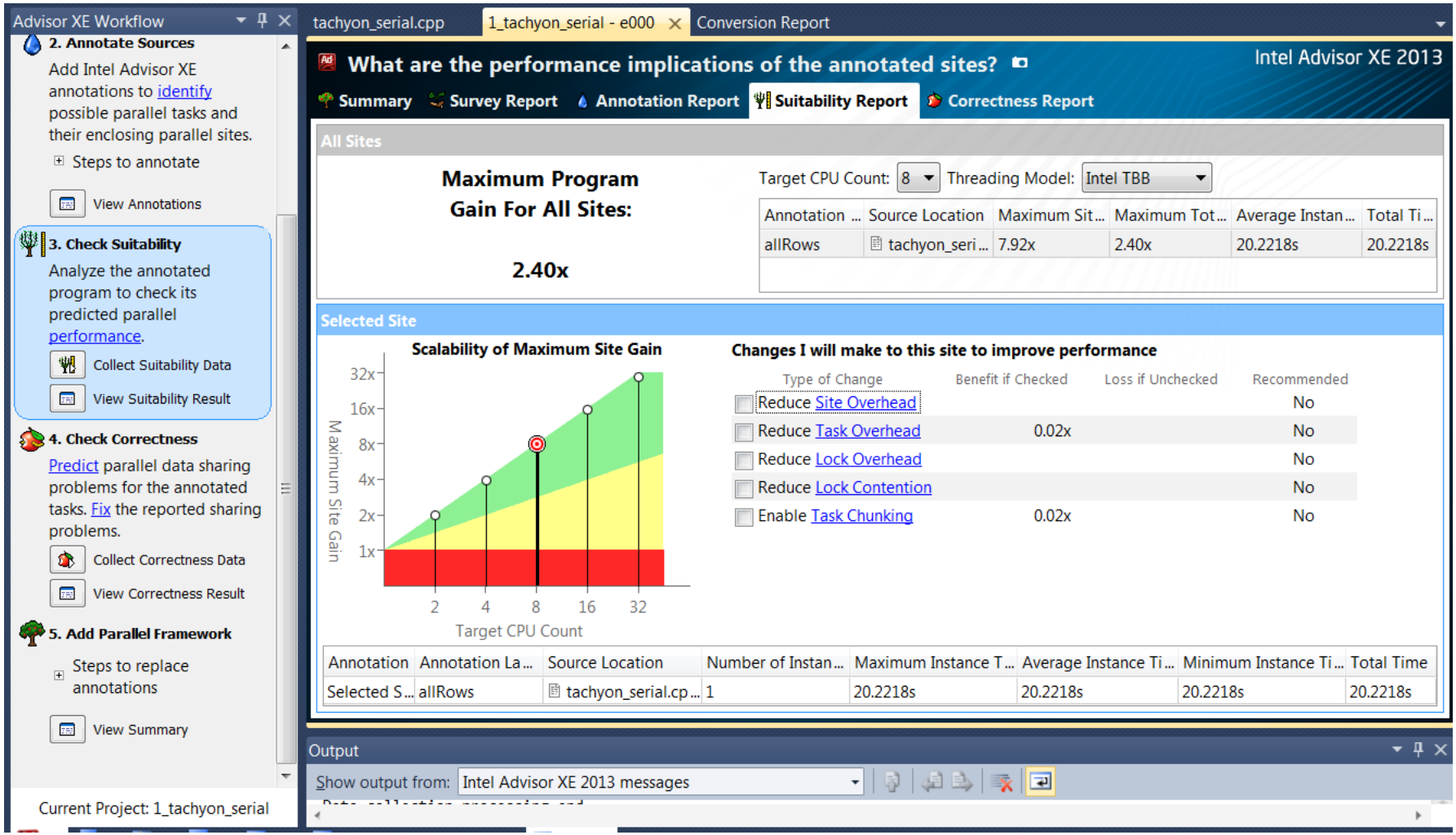
Annotate Sources using Survey Data

Function Call Sites and Loops	Total Time %	Total Time
rt_renderscene	96.2%	19.9664s
renderscene	96.2%	19.9662s
trace_region	96.2%	19.9662s
trace_shm	96.2%	19.9662s
thread_trace	96.2%	19.9662s
parallel_thread [loop]	96.2%	19.9662s
parallel_thread [loop]	96.1%	19.9462s
parallel_thread	96.0%	19.9262s
render_one_pixel	93.1%	19.3208s
trace	79.2%	16.4369s
shader	43.6%	9.0521s
shader [loop]	35.4%	7.3348s

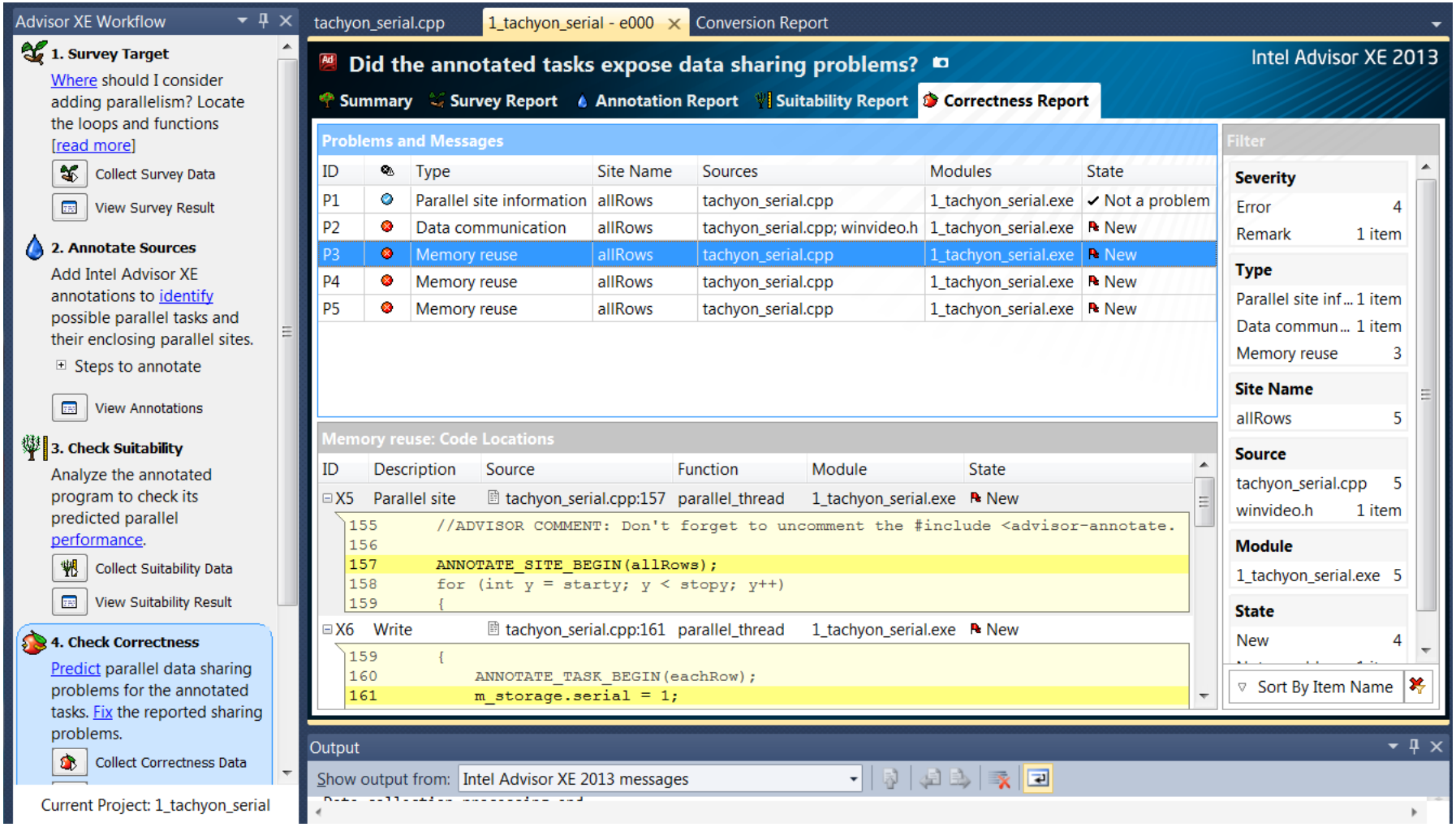
```
ANNOTATE_SITE_BEGIN(allRows);
for (int y = starty; y < stopy; y++)
{
    ANNOTATE_TASK_BEGIN(eachRow);
    m_storage.serial = 1;
    m_storage.mboxsize = sizeof(unsi
    m_storage.local_mbox = (unsigned
    memset(m_storage.local_mbox,0,m_

    drawing_area drawing(startx, tot
    for (int x = startx; x < stopx;
        color_t c = render_one_pixel
        drawing.put_pixel(c);
    }

    if(!video->next_frame())
    {
        free(m_storage.local_mbox);
        return;
    }
    free(m_storage.local_mbox);
    ANNOTATE_TASK_END(eachRow);
}
ANNOTATE_SITE_END(allRows);
```

Pick out expensive loop and add annotations

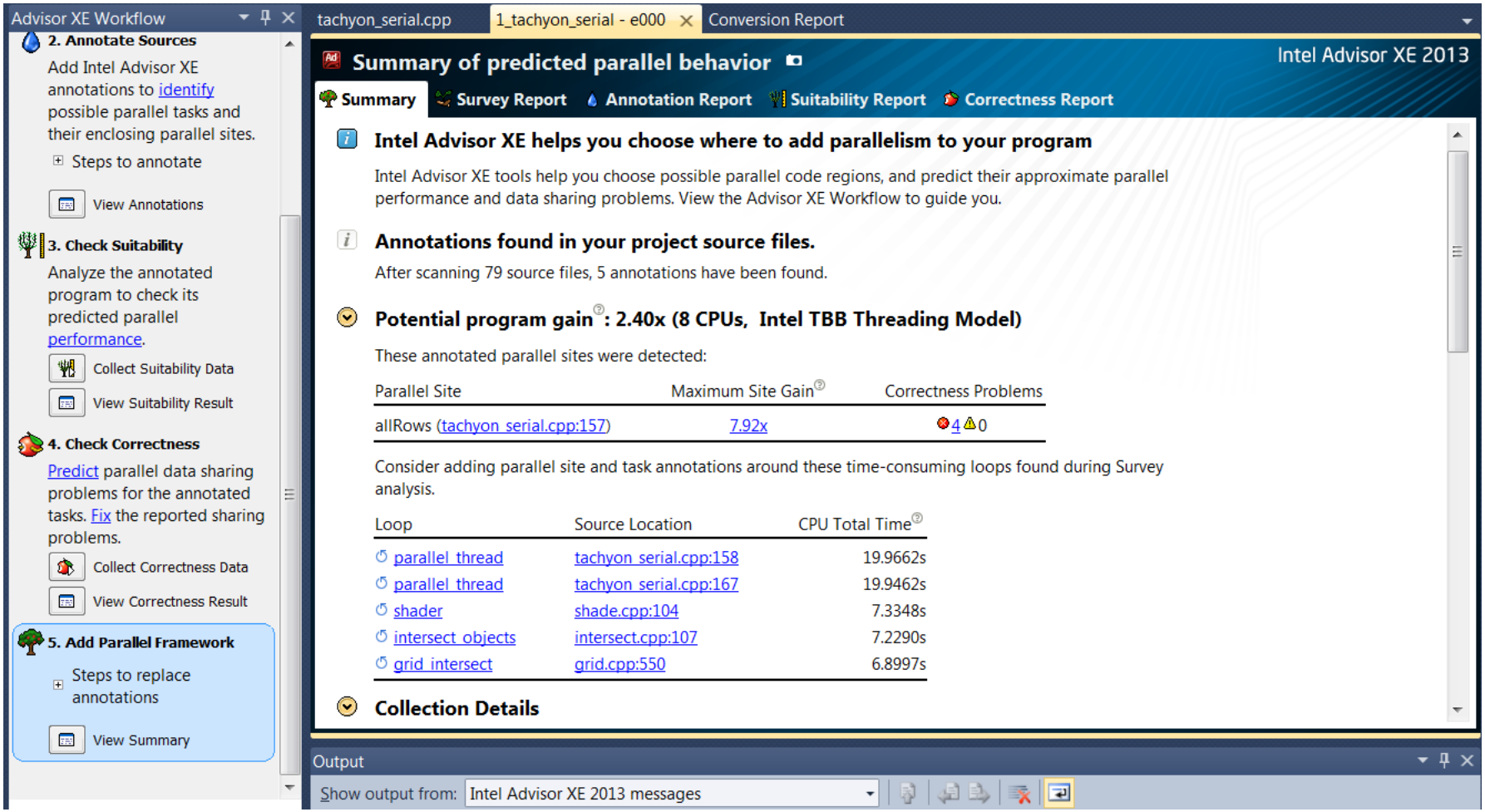
Check Suitability



Check Correctness



Add Parallel Framework



Add Parallel Framework

How can Intel Advisor help?

- Contains overhead metrics for selected models
- Allows to play with some implementation options
- Detailed help pages for different frameworks.

Serial Code with Intel Advisor Annotations	Parallel Code using Intel TBB

`// Locking`
`ANNOTATE_LOCK_ACQUIRE();`
`  Body();`
`ANNOTATE_LOCK_RELEASE():`
	`// Locking can use various mutex types provided`
	`// by Intel TBB. For example:`
	`#include <tbb/tbb.h>`
	`...`
	`tbb::mutex g_Mutex;`
	`...`
	`{`
	`  tbb::mutex::scoped_lock lock(g_Mutex);`
	`  Body();`
	`}`

`// Do-All Counted loops, one task`
`ANNOTATE_SITE_BEGIN(site);`
`  For (I = 0; I < N; ++) {`
`  ANNOTATE_ITERATION_TASK(task);`
`  {statement;}`
`  }`
`ANNOTATE_SITE_END();`
	`// Do-All Counted loops, using lambda expressions`
	`#include <tbb/tbb.h>`
	`...`
	`tbb::parallel_for(0,N,[&](int I) {`
	`  statement;`
	`});`

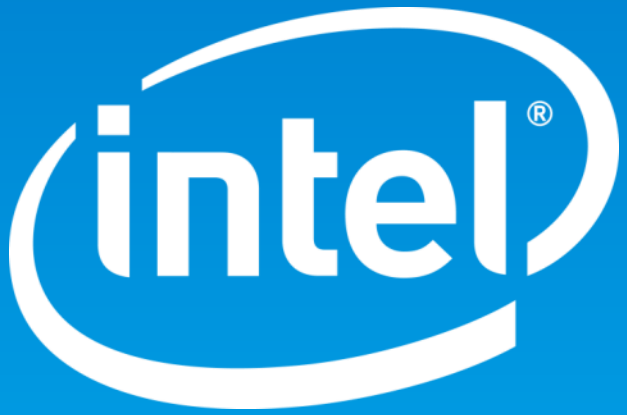
In Summary

Advisor XE is a tool to assist in the parallelization of serial code.

It helps the developer/architect by

- Finding the sections of code taking up most of the time
- Modeling the available speedup for parallelizing those sections of code.
- Finding correctness errors that will occur if the section is naively parallelized

It does all this using a lightweight annotation system to allow quickly trying multiple code change possibilities without choosing a parallel framework.


intel

Legal Disclaimer & Optimization Notice

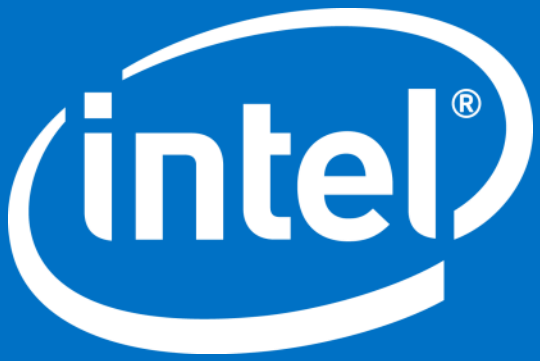
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel®